





PYTHON







Python
is the name
of a programming
language
as well

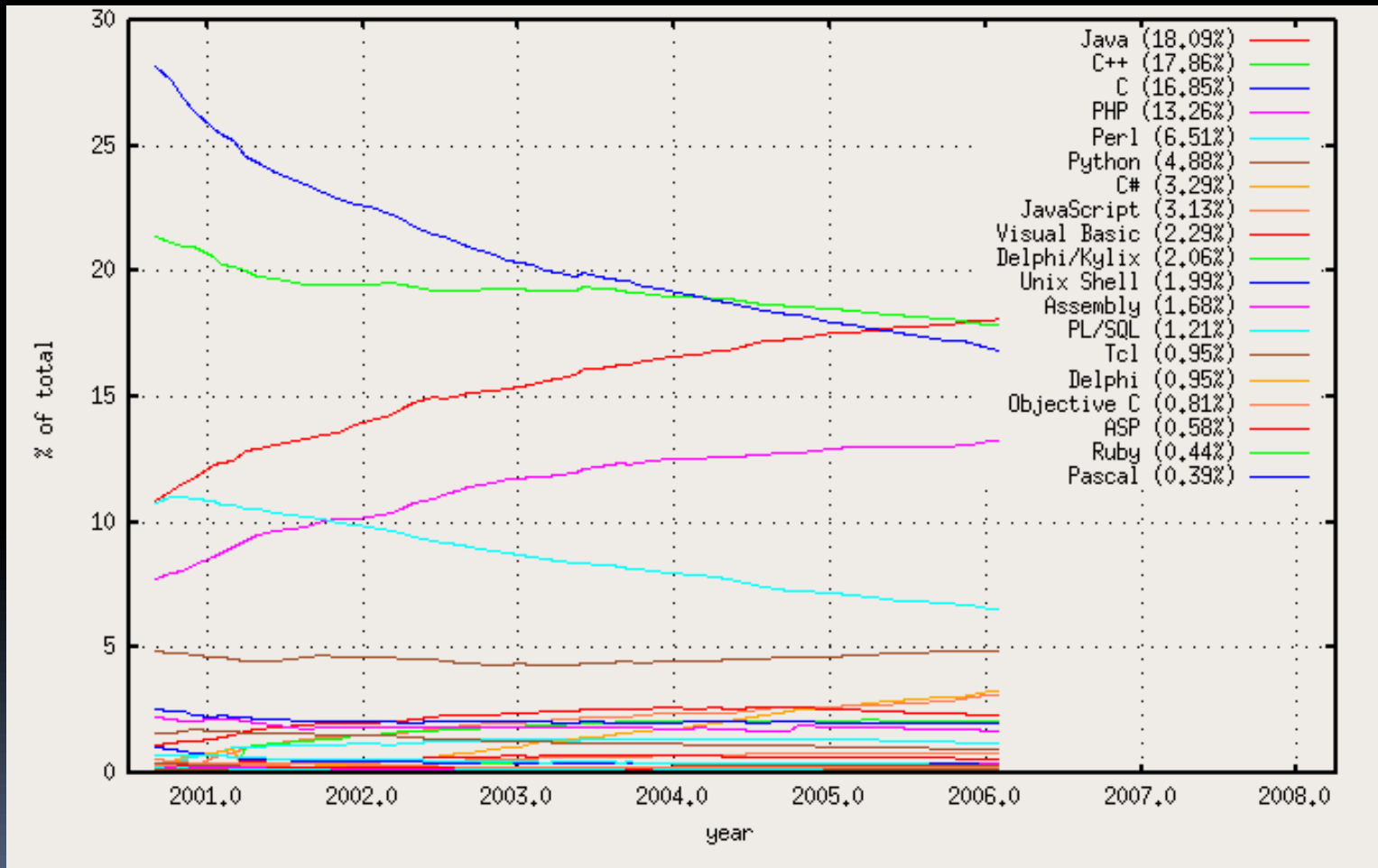
Programming Languages Usage

Position	Language	Ratings
1	Java	19.1%
2	C	15.2%
3	C++	10.1%
4	PHP	8.7%
5	Visual Basic	8.4%
6	Perl	6.2%
7	Python	3.8%
8	C#	3.7%
9	JavaScript	3.1%
10	Ruby	2.6%

The numbers are from May 2007.

www.zetcode.com/wxpython/introduction

Programming Languages Usage





Python

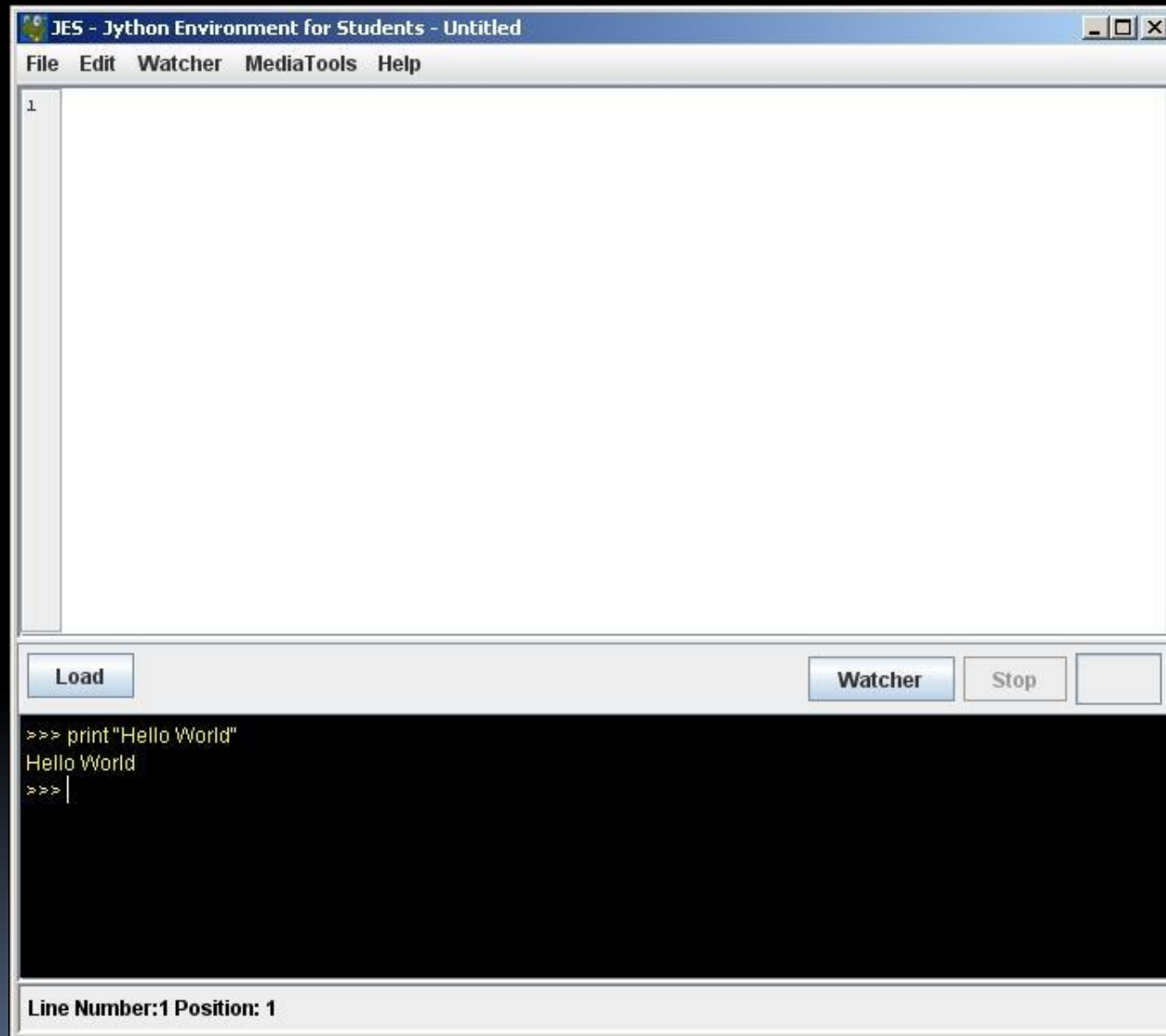
- Open Source (Free)
- Portable
- Simple Syntax
- Well Documented
- www.python.org

Programming with Python

- Data
 - Constants and Variables
- Instructions (Code)
 - Assignment (=)
 - If .. Then statement
 - Loops

.....

JES



Variables

Variables are names for data

Examples of good variable names:

age = 99

province = "Alberta"

x = 15

y = 32

gasprice = 0.95

pi = 3.14

Variables (names)

Examples of incorrect variable names:

- `76trombones = "big parade"`

Not good because the variable name starts with a number .

- `more$ = 1000000`

Not good because the variable name has the dollar sign.

- `class = "Computer Science 203"`

Not good because the variable name used: 'class' is a reserved word

Variables (Data Types)

- Numbers:

integers (0, 12, 17, -20,...),

floats (0.0, 3.14, 6.02e23,.....)

- Operators:

+ - * / % (modulus or remainder) ** (exponentiation)

- Strings:

“anything in double quotes”

Variables (Data Types)

- `type("Hello World") : <type 'str'>`
- `type(10) : <type 'int'>`

Issues:

```
print 1,000,000 : 1 0 0
```


Variables (names)

Reserved words

and - assert - break - class - continue - def
del - elif - else - except - exec - finally
for - from - global - if - import - in
is - lambda - not - or - pass - print
raise - return - try - while - yield



Some practice

- Create a variable call it var1 and store the number 9 in it.
 - Create a second variable call it var2 and store the number 1 in it.
 - Get the sum of var1 and var2 and store the result in var3.
 - Switch the values of var1 and var2.
- 

Very Important

Indentation is extremely important.

Example :

while {condition that the loop continues}:

 {what to do in the loop}

 {have it indented, usually four spaces}

{the code here is not looped}

{because it isn't indented}

Python is case sensitive

Variable names:

- **X != x** (X is different from x)
- **VariableName != variableName**

Function names

- **Print != print**

if statement

```
if {conditions to be met}:  
    {do this}  
    {and this}  
    {and this}  
{but this happens regardless}  
{because it isn't indented}
```

Example 01

```
y = 1  
if y == 1:  
    print "y still equals 1, I was just checking"
```


Comparison operators

- == equal
- >= grater than or equal
- <= less than or equal
- < less than
- > grater than
- != different

if statement

Example02

```
a = 1
if a > 5:
    print "This shouldn't happen."
else:
    print 'This should happen.'
```

Example03

```
z = 4
if z > 70:
    print "Something is very wrong"
elif z < 7:
    print "This is normal"
```

Loops – for loop

for item **in** container:

action to repeat for each item in the container

else:

action to take once we have finished the loop.

Example01

```
sum = 0
```

```
for x in [1, 2, 3, 4]:
```

```
    sum = sum + x
```

Loops – for loop

Example02

```
newList = [45, 'eat me', 90210, 'The day has come', -67]  
for value in newList:  
    print value
```

Loops - while loop

while {condition that the loop continues}:

 {what to do in the loop}

 {have it indented, usually four spaces}

{the code here is not looped}

{because it isn't indented}

Example 01

```
x = 10
```

```
while x != 0:
```

```
    print x
```

```
    x = x - 1
```

```
    print "wow, we have counted x down, and now it equals", x  
print "And now the loop has ended."
```

Loops - while loop

Example02

```
print "We will show the even numbers up to 20"
```

```
n = 1
```

```
while n <= 20:
```

```
    if n % 2 == 0:
```

```
        print n
```

```
    n = n + 1
```

```
print "there, done."
```

Important notes

A condition in an if statement or a while loop can be a simple one:

expression **comparison Operator** expression

if (age > 100):
if (x%2 == 0):

while((i + j) < (k * l - m)):

Or a mix of several simple conditions:

logical operator (condition₁) **logical operator** (condition₂)
.....**logical operator**(condition_n)

if (age>100 **and** salary>100000):
if(color == "red" **or** color=="pink"):

The difference between for & while

```
x = 10
```

```
while x != 0:
```

```
    print x
```

```
    x = x - 1
```

#we have to decrement/increment the counter

```
sum = 0
```

```
for x in [1, 2, 3, 4]:
```

```
    sum = sum + x
```

#the for looks for the next value of x automatically

functions

- **Built-in functions.**

`print` "something", somethingelse , ...

Manipulating pictures:

Get a picture from the internet - Google : [free pictures rockies](#)

`makePicture, show, getColor, SetColor,.....`

```
picture = makePicture("C:\\Documents and Settings\\aguerbas\\Desktop\\python.jpg")
```


```
show(picture)
```

Builtin function manipulating pictures

```
fileName = pickAFile()  
picture = makePicture(fileName)  
show(picture)
```

```
picture = makePicture(pickAFile())  
show(picture)
```

```
# Creates an empty picture 200 pixels wide by 100 pixels high  
emptyPicture = makeEmptyPicture(200, 100)
```



Builtin functions manipulating pictures

- Let's do the wiki examples
- 

2D Images – Bitmap images

- Bitmap-based images are comprised of pixels in a grid. Each pixel in the image contains information about the color to be displayed.

3 colors (RGB) are used

Each color can have a value from : 0 to 255

- Example :
(0,0,0) black pixel
(255,255,255) white pixel

Builtin functions manipulating pictures

#Increasing the red component by 150% for all pixels of the picture:

```
picture=makePicture(pickAFile())
for p in getPixels(picture):
    value=getRed(p)
    setRed(p,value*1.5)
show(picture)
```

#Converting to grayscale:

```
picture=makePicture(pickAFile())
for p in getPixels(picture):
    redComponent = getRed(p)
    greenComponent = getGreen(p)
    blueComponent = getBlue(p)
    avg = (redComponent + greenComponent + blueComponent)/3
    setColor(p,makeColor(avg,avg,avg))
show(picture)
```

User Defined Functions

function with no parameters and no returned value

```
def hello():
```

```
    print "Hello World!"
```

When you call this function using the interactive console like this:

```
hello()
```

It will print Hello World!

Simplest Arguments

```
def withParametres ( i, n, txt ):
```

```
    i = 0
```

```
    while i < n:
```

```
        print txt
```

```
        i = i + 1
```

When you call this function using the interactive console like this:

```
withParameters(1,10,"Hello")
```

This function will print the word Hello 10 times.

User Defined Functions

function with parameters and with returned value

```
def product (number1,number2):  
    result = number1 * number2  
return result
```

The value returned by a function can be stored in a variable like this:

```
variableName = product(2,3)
```

The previous instruction can be in the same file where the function product is defined or can be typed in the interactive (black) console

The following code shows how to create a function and how to call it
The code has to be in the same file.

#This is the beginning of the code

#create a function called convertToNegative and it has one parameter called picture

def convertToNegative(picture):

for p in getPixels(picture):

redComponent = getRed(p)

greenComponent = getGreen(p)

blueComponent = getBlue(p)

setRed(p,255-redComponent)

setGreen(p,255-greenComponent)

setBlue(p,255-blueComponent)

show(picture)

#select a picture you want to convert

picture=makePicture(pickAFile())

#call the function that will do the job of converting the picture and showing it
convertToNegative(picture)

#This is the end of the code